Project information

Created on
October 29, 2024

Merge branch 'dev' into 'main'
Matthias Hersche authored 3 hours ago

Name	Last commit	Last update
backend	Dev	3 hours ago
contracts	added basic backend functions and prettier	1 week ago
frontend	Dev	3 hours ago
.gitignore	added basic backend	3 weeks ago
.prettierrc	added basic backend functions and prettier	1 week ago
Caddyfile	small improvements	6 days ago
README.md	Dev	3 hours ago
docker-compose.yml	small improvements	6 days ago

README.md

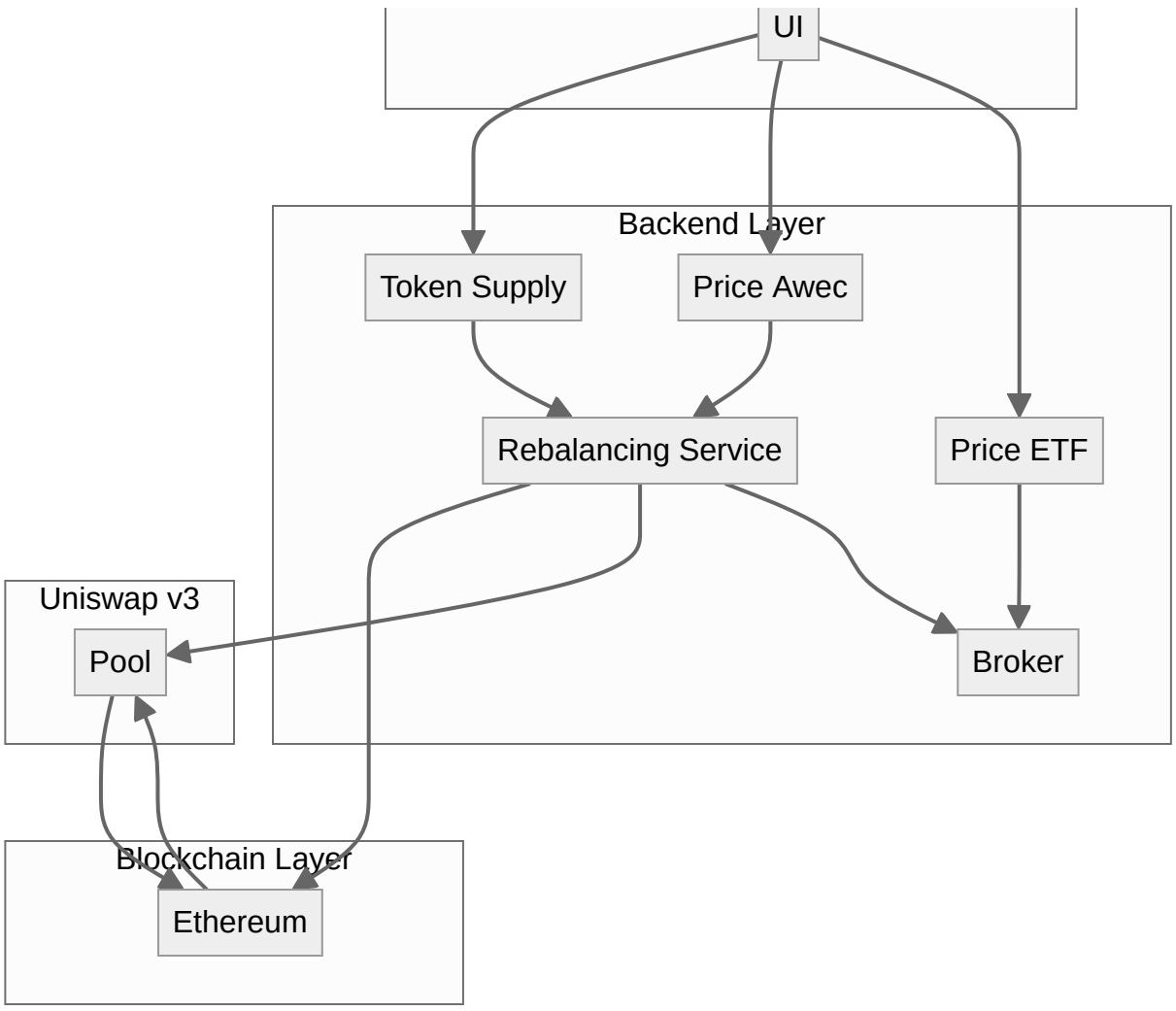
Overview

The All World ETF Coin (AWEC) is an ERC-20 token implementation that creates a bridge between traditional ETF markets and the ethereum ecosystem. It tracks the Invesco FTSE All-World UCITS ETF (ISIN: IE000716YHJ7) and maintains price stability through an automated balancing mechanism.

Network: Sepolia Testnet

- USDC Contract
- USDC Token
- AWEC Contract
- AWEC Token
- AWEC/USDC Pool

System Architecture



Frontend Layer

The frontend of our Ethereum Stablecoin project is a React-based web application that provides a user-friendly interface for monitoring the AWEC stablecoin metrics and includes links to Uniswap.

Components

1. Real-Time Price Displays

- **ETF Price Display**: Shows the price of the underlying ETF
- **AWEC Pool Price Display**: Displays the current price of AWEC in the liquidity pool
- **Supply Information**: Shows the total circulating supply of AWEC tokens

2. Trading Interface

- Buttons, that redirect to the Uniswap website for buying and selling AWEC
- The routing to Uniswap includes pre-filled token addresses

3. Technical Implementation

- Built with React 18
- Containerized using Docker and Nginx

Backend Layer

The backend of our Ethereum Stablecoin project is based on Node.js, built with Express and TypeScript. It handles the core logic for price tracking, pool management, and stablecoin balancing. It also serves as an API service for the frontend.

Components

1. Core Services

ETF Price Service

- Fetches ETF price data from external API
- Provides price information for the stablecoin pegging mechanism

AWEC Service

- Provides the total supply of AWEC tokens
- Provides pool price information
- Manages the rebalancing

Balancing Service

Broker Service

- Simulates ETF trading operations
- Manages ETF counting and tracking

2. API Endpoints

```
GET / api / price / etf; // Returns current ETF price
GET / api / price / awec; // Returns current AWEC pool price
GET / api / supply; // Returns total AWEC supply
```

3. Smart Contract Integration

- Interfaces with Ethereum smart contracts using ethers.js
- Supports both read and write operations

4. Technical Implementation

- Built with Express.js and TypeScript
- Uses Uniswap V3 SDK for pool management

Blockchain Layer

Our Ethereum stablecoin project (AWEC - All-World ETF Coin) is built with OpenZeppelin's smart contract libraries. It implements a secure and standard-compliant ERC-20 token.

Token Implementation

- Implements ERC-20 standard interface
- Name: "All-World ETF Coin"
- Symbol: "AWEC"
- Decimals: 18 (default)

Key Functions

For Token Holders

- `transfer(address to, uint256 amount)`: Transfer tokens to another address
- `approve(address spender, uint256 amount)`: Approve another address to spend tokens
- `transferFrom(address from, address to, uint256 amount)`: Transfer tokens on behalf of another address

Owner-Only Functions

- `mint(address to, uint256 amount)`: Create new tokens
- `burn(uint256 amount)`: Destroy tokens
- `pause()`: Pause all token transfers
- `unpause()`: Resume token transfers

Security Features

- Pausable functionality for emergency stops
- Burnable tokens with owner restrictions

Error Messages

- `AllWorldEtfCoin_MustBeMoreThanZero()`: Amount must be greater than zero **What it means:** The amount you're trying to use in the operation is zero or negative.
- `AllWorldEtfCoin_BurnAmountExceedsBalance()`: Cannot burn more than available balance **What it means:** You're trying to burn more tokens than you actually have.
- `AllWorldEtfCoin_NotZeroAddress()`: Cannot mint to zero address **What it means:** You're trying to interact with the zero address (0x0000000000000000000000000000000000000000).

Supportive Infrastructure

Docker

Our application uses a multi-container setup with Docker:

- **Caddy Container**: Reverse proxy and SSL handler

Caddy Implementation

Caddy serves as our reverse proxy, handling routing between frontend and backend:

Nginx Configuration (Frontend)

Nginx serves our static React files and handles client-side routing:

- Serves static files efficiently
- Handles fallback errors

Price Stability Mechanism

A critical feature of our system is to ensure that the price of the AWEC tokens remains stable in relation to the price of the ETF. The Rebalancing Job runs in a time interval consists of two rebalancing mechanisms.

Overview

This mechanism is implemented in the `UniV3BalancingService` class, which interacts with the necessary services and contracts to achieve price stability.

The `UniV3BalancingService` class has two main responsibilities:

1. **Rebalancing the Pool**: Adjusting the amount of AWEC and USDC in the Uniswap V3 pool to maintain the desired price.
2. **Rebalancing Total Supply**: Ensuring that the total supply of AWEC tokens matches the number of ETFs held by the broker.

Key Components

- **Uniswap V3 Pool**: The decentralized exchange pool where AWEC and USDC are traded.
- **Broker Service**: Manages the buying and selling of ETFs.
- **AWEC Token**: The token whose price is stabilized.
- **USDC Token**: The stablecoin used for trading AWEC.

Workflow

1. **Rebalance Job**: The main function that organizes the rebalancing process. It runs on a regular basis to ensure continuous price stability.
2. **Rebalance Pool**: Adjusts the pool's liquidity by buying or selling AWEC tokens based on the price difference between the current pool price and the ETF price.
3. **Rebalance Total Supply**: Ensures that the total supply of AWEC tokens matches the number of ETFs held by the us.

Detailed Explanation

Rebalance Job

The `rebalanceJob` function is called periodically (every minute). It initiates the rebalancing of the pool and the total supply.

Rebalance Pool

1. **Fetch ETF Price**: The current ETF price is fetched.
2. **Get Pool State**: The current state of the Uniswap V3 pool is retrieved.
3. **Calculate Price Difference**: The difference between the ETF price and the current pool price is calculated.
4. **Threshold Check**: If the price difference exceeds the threshold (0.01 USD), the system proceeds to rebalance the pool.
5. **Calculate Required Amount**: The amount of AWEC tokens needed to adjust the pool is calculated.
6. **Adjust Pool Liquidity**:
    - **If AWEC Price is Higher**: The system mints AWEC tokens and sell them to the pool until the price matches the ETF price.
    - **If AWEC Price is Lower**: The system buys AWEC tokens from the pool until the price matches the ETF price.

Rebalance Total Supply

1. **Fetch ETF Count**: The number of ETFs held by the broker is fetched.
2. **Get Total Supply**: The total supply of AWEC tokens is retrieved.
3. **Calculate Difference**: The difference between the ETF count and the total supply is calculated.
4. **Mint or Burn AWEC Tokens**:
    - **If ETF Count is Greater**: The system mints additional AWEC tokens.
    - **If ETF Count is Less**: The system burns excess AWEC tokens.

Examples

- **Current ETF Price**: $1.00
- **Current AWEC Price**: $1.05
- **Price Difference**: $0.05
- **Threshold**: $0.01

Since the price difference exceeds the threshold, the system will sell AWEC tokens to lower the price. The system calculates the amount of AWEC tokens to sell and executes the necessary swaps so the price gets down to the level of the ETF price.

Example 2: AWEC Price is Lower Than ETF Price

- **Current ETF Price**: $1.00
- **Current AWEC Price**: $0.95
- **Price Difference**: $0.05
- **Threshold**: $0.01

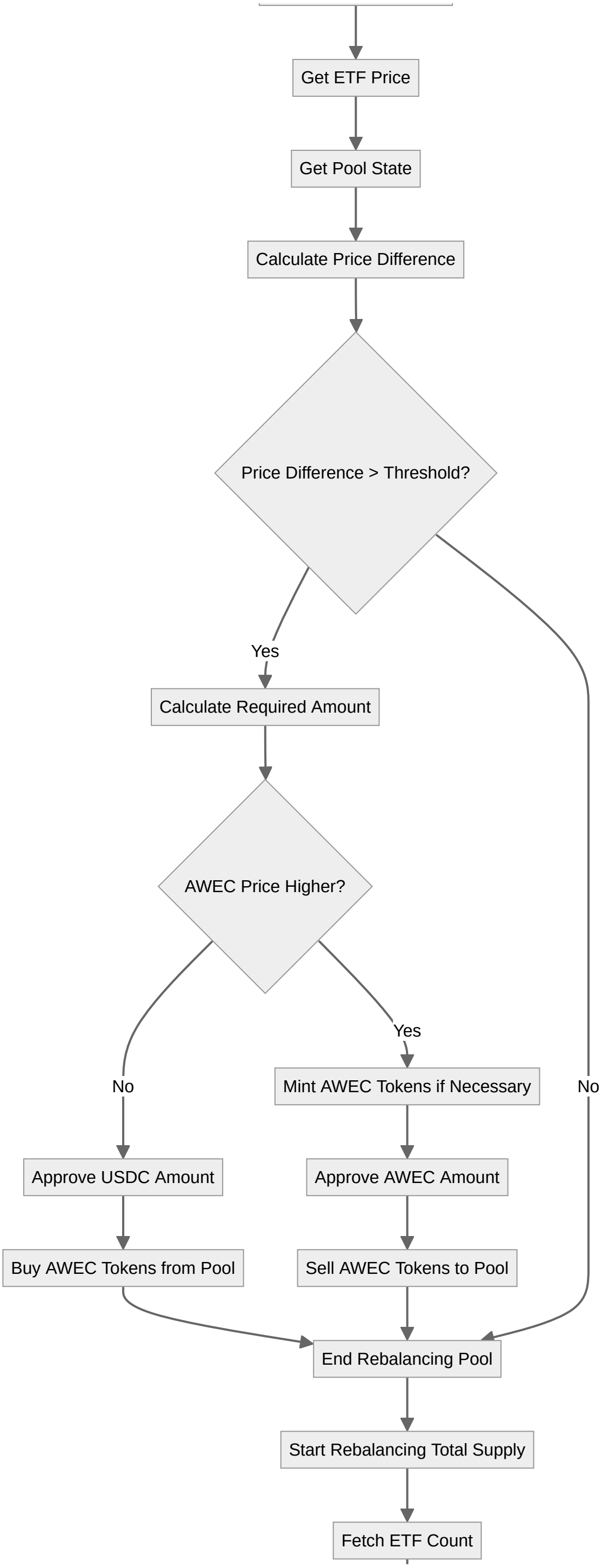
Since the price difference exceeds the threshold, the system will buy AWEC tokens to raise the price. The system calculates the amount of AWEC tokens to buy and executes the necessary swaps so the price of AWEC rises until it matches the price of the ETF.

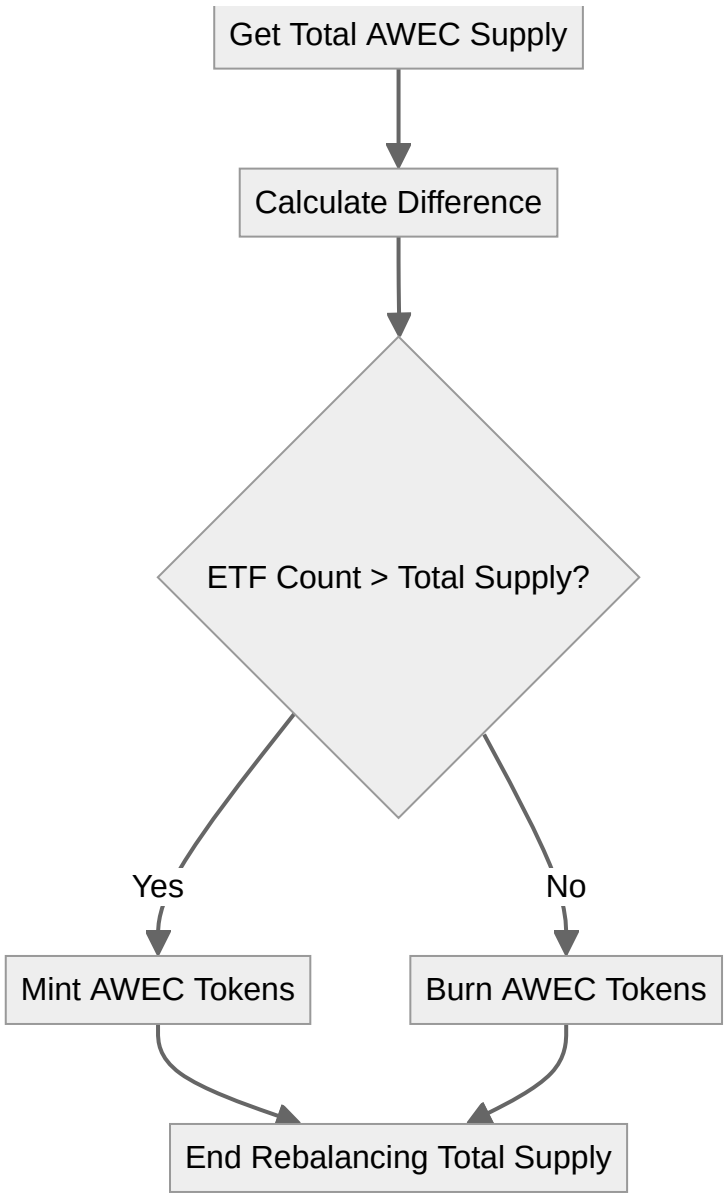
Example 3: Rebalancing Total Supply

- **ETF Count**: 1000
- **Total Supply of AWEC**: 950
- **Difference**: 50

Since the ETF count is greater than the total supply, the system will mint 50 additional AWEC tokens to match the ETF count.

Chart


Get ETF Price
Get Pool State
Calculate Price Difference
Price Difference > Threshold?
Yes
No
Calculate Required Amount
AWEC Price Higher?
Yes
No
Mint AWEC Tokens if Necessary
No
Approve USDC Amount
Approve AWEC Amount
Buy AWEC Tokens from Pool
Sell AWEC Tokens to Pool
End Rebalancing Pool
Start Rebalancing Total Supply
Fetch ETF Count


Get Total AWEC Supply
Calculate Difference
ETF Count > Total Supply?
Yes
No
Mint AWEC Tokens
Burn AWEC Tokens
End Rebalancing Total Supply